Cloud-based framework for simulation-based optimisation of ship energy systems

Lappalainen, Jari TJ

University of Strathclyde Publishing
2019

Lappalainen , J TJ , Korvola , T , Nurminen , J K , Lepistö , V & Mäki-Jouppila , T 2019 , Cloud-based framework for simulation-based optimisation of ship energy systems . in G Theotokatos & A Coraddu (eds) , Proceedings of the 2nd International Conference on Modelling and Optimisation of Ship Energy Systems (MOSES2019) . University of Strathclyde Publishing , Glasgow , pp. 65-71 , International Conference on Modelling and Optimisation of Ship Energy Systems , 08/05/2019 . < http://130.159.17.33/MOSES2019/ >

http://hdl.handle.net/10138/307625

MOSES2019-02011

Cloud-based framework for simulation-based optimisation of ship energy systems

Jari Lappalainen[a*], Timo Korvola[a], Jukka K. Nurminen[a], Vesa Lepistö[b] and Tero Mäki-Jouppila[b]

[a] VTT Technical Research Centre of Finland Ltd, Espoo, Finland
[b] Meyer Turku Oy, Turku, Finland
* Corresponding Author jari.lappalainen@vtt.fi

Abstract

Easy and affordable access to massive computing capacity in the cloud provides new opportunities to take advantage of simulations. In this paper, we discuss simulation-based optimization and, in particular, the practical challenges of applying it in today's cloud computing environments. Our focus is the engineering and operation of energy-efficient ships. We report our experiences in designing and developing a framework for simulation-based optimization in the cloud environment. Starting from a set of hypotheses, we discuss how our practical experience in building the system confirms or refutes those hypotheses, what kind of practical solutions we have come up with, and what kind of future needs we have detected.

Keywords: Optimization, evolutionary algorithms, cloud computing, dynamic simulation, energy efficiency.

1 INTRODUCTION

A proper dynamic simulation model provides a tool for evaluating different energy management concepts in the design phase of a system, and the important aspects of energy efficiency and system's operability can be simultaneously addressed. The same simulation approach can be used for advancing energy efficiency in operating ships as well. As dynamic simulation mimics the system behaviour including the interactions with the automation system, it inherently fits in the concept of digital twins which has recently gained large attention [1].

Incorporating the dynamic system simulation method into engineering workflow includes two major prerequisites. Firstly, good modelling tools and engineers skilled in them are needed. Secondly, the models should be efficiently and fully exploited. While the first part is progressing widely in the marine industry, the second part deserves more attention. Efficient means to exploit the developed models are crucial for increasing the impact of dynamic system simulation.

Incorporating optimization with the simulation models is one direction to search for improved efficiency and quality in simulation aided engineering. This approach is known as simulation-based optimization [2,3] or simply simulation optimization [4–6]. The approach is especially popular for optimization in stochastic context. Regarding dynamic simulation, optimization can address both system design and operative decisions such as concept level strategic choices, equipment dimensioning, tuning of process automation parameters, or operation with the existing equipment (for an overview of applying modelling and optimizing ship energy systems see e.g. [7].

Our framework aims at enabling optimization for dynamic models not originally intended for the purpose. The models should be considered as black box simulators. Thus we have chosen evolutionary algorithms as the primary optimization approach. This also allows parallel computing taking advantage of even hundreds of computing cores affordably available in the cloud.

In this paper, we share experiences of ongoing research to develop methodology for agile utilization of optimization in the cloud to support decision making in complex maritime problems involving energy systems and operative aspects. We present the architecture of the optimization platform, which employs evolutionary algorithms and cloud computing for dynamic simulators. We discuss the challenges we have met when implementing the system in practise.

In the following, we introduce the hypotheses that were the starting points in our work. In section 6, we will look back to the hypotheses to see how correct our assumptions were. The overarching hypothesis is that *today's cloud computing infrastructure enables an easy and flexible framework for taking out-of-a-box marine simulators and solving related optimization problems*. We divide this further into a set of sub-hypotheses:

1. Engineering problems that arise within the in-house simulation activities can commonly be converted into simulation-based optimization form, without reformulation of the simulation model (i.e. using them as black box simulators).
2. The black box simulators can be transferred and run in the cloud in the form of containers.
3. Harnessing computing capacity in the cloud allows use of gradient-free optimization algorithms. Computation efficiency is achieved

by parallel computing which the cloud provides on demand.
4. Evolutionary optimization algorithms provide a means to find adequate solutions to the engineering problems.
5. Machine learning can help in directing the search of the optimization algorithms to improve the performance.
6. The optimization framework allows changing the simulation model or the simulator, goals and constraints of the optimization, and the optimization algorithm. It allows the use of simulators built for different operating systems (especially Windows and Linux).
7. The optimization framework promotes the optimization-as-a-service approach.

The rest of the paper is structured as follows. In section 2, we discuss challenges and experiences of using simulation in maritime field. Then, we review the literature on simulation-based optimization, especially from the cloud computing point of view. Section 4 describes our implementation using Docker, Kubernetes and Dask. Section 5 introduces our first industrial simulation case. In section 6, we share our experiences and lessons learned in the implementation. Finally, in section 7, we conclude and present topics for future work.

2 SYSTEM SIMULATION IN THE MARITIME FIELD

To fulfil all the new requirements the marine engineering is facing, there is a great demand for proper tools to examine and verify alternative engineering solutions for the ship energy management. In comparison to industrial thermal systems onshore, the ship design has numerous specific characteristics and constraints [8]. For example, the ships need to operate on a large load range and the load variations are much more frequent than in the traditional power plants. The requirements for system reliability and availability are higher than in conventional power industry. There is limited space for equipment placing and service on board, and the weight (carried over on every cruise for the vessel's lifetime) of the systems is meaningful too.

The ship route and operation area are essential factors in the ship design. The targeted operation area sets the main constraints, but on the other hand, a cruising route may include an extensive variety of different conditions. Regarding the optimization of ship energy systems, it is relevant to ask, how tight an optimization is reasonable. Too tightly optimized design may reduce system's flexibility. In this respect, it helps if the models used within the optimization take the operability aspects into account. Even then a risk exists, because there can be significant changes in the operation profile and conditions during the vessel's lifetime. If the optimization is focusing on very narrow domains, such as a diesel engine only, a risk for sub-optimal solution is high. System-wide models and simulations help to overcome these problems. The number of system-wide, marine related modelling literature is still comparatively limited, but it is increasing fast, driven by the fact that fulfilling the new and future vessel requirements demands the holistic approach. The essential goal is to design and evaluate advanced process and control solutions. Naturally, this can be approached from many different angles. For example, [9] presented a system level approach using Matlab-Simscape environment to systematically model marine engines' energy flow including the main energy producers and consumers in a cruise ship. Lepistö et al. [8] investigated waste heat recovery (WHRS) and chilled water (CWS) systems with system-wide modelling and dynamic process simulation. They investigated novel energy solutions such as Alaska cooler, heat accumulator tanks, LNG cold recovery system and low temperature energy conversion to electricity. Lampe et al. [10] presented a method where they combined thermodynamics and analysis of the overall availability of the energy systems. They addressed different WHRS arrangements.

The questions around energy efficiency are multifaceted, and thus the steady-state modelling seems to be the most commonly used approach. When facing system level complexity, one useful tendency is to exploit independent developments, i.e., combine simulation models. An example of this is found in [11], where distributed engine control system and thermodynamic models are co-simulated. The co-simulation systems typically get too complicated to allow mathematical optimization studies, whereas they have an important role in automation testing and operator training. The control and operation aspects, which dynamic simulation is inherently considering, are crucially important when novel concepts are addressed. Furthermore, since a rigorous simulation model is capable to mimic its real counterpart, it is considered as an inherent part of a Digital Twin. This makes system-wide modelling and dynamic simulation very interesting for the operating ships as well.

Whenever a proper simulator exists, it should be thoroughly exploited. A leading tool in this front is simulation-based optimization, an approach, where a simulator is applied in the optimization loop.

3 FROM SIMULATION MODELS TO SIMULATION-BASED OPTIMIZATION

The principle of repeatedly varying parameter values and seeing their influence via simulation is old. The classic way is to execute numerous simulations with incremental parameter changes, or to exploit some design of experiments technique. The approach is straightforward and in many cases capable to produce a crisp answer for the problem, see for instance the study of thermo-economic optimization of waste heat recovery from large diesel engine by ORC with five different working fluids [12]. However, the problem size and complexity, and computationally expensive simulation are factors that easily make the brute force optimization unfeasible. On the other hand, it is typically far easier to create a simulation model than a corresponding optimization model to be used with classical optimization methods. The recent increase in computational capacity and some algorithmic developments have made simulation-based optimization increasingly popular (see [2,13] for an overview). It is particularly useful in cases where creating an optimization model (e.g. mathematical optimization or

dynamic programming) is demanding. This is common in industrial problems, and in problems, which involve stochastic behaviour. Simulation-based optimization has been applied e.g. to optimization of building performance [14]. In the maritime domain, [15] investigated multi-objective optimization with respect to environmental and economic objectives and with considerations of operational and regulatory requirements during the ship lifetime. Sakalis and Frangopoulos [16] presented a general method for synthesis, design and operation optimization of ship energy systems where the triple optimization (synthesis, design, operation) is performed on a single level. They used genetic algorithms (GA).

Regarding simulation-based optimization in industrial use, the most straightforward way is to treat the simulation model as a black box. The other alternative is that the simulation model provides gradient information to the optimization layer. This would allow the use of gradient-based optimization methods, where more focused search would save computational effort by reducing the number of times the model needs to be simulated. However, getting reliable gradient information from complex models is difficult [2]. A further option is automatic differentiation, which is able to compute derivatives via analysis of the arithmetic operations in the program code [17]. However, it is not able to deal with complex simulation models. Moreover, in the energy system optimization where operational aspects are included, we are often interested in time series. This makes the use of these more sophisticated methods hard. Another type of approach to save computing is to derive simpler surrogate models from the expensive simulation models [18,19].

In the industrial problems, dynamic models are often large and complex. They also evolve over time, as consistent model updates and re-use are crucial for the cost-efficiency in simulation aided engineering. Therefore converting them into classical optimization problems is very challenging in the tight time frames of engineering projects. Evolutionary algorithms, such as GA, have been widely used for optimizing within such problems, where the simulation model can be characterized as black box. In other words, the model provides limited or no information about its internal state, e.g. derivatives of the state variables. Like the name reveals, GA builds on the idea of natural evolution. In this context it means that a set of simulations (a population) is executed with certain decision parameter values (genes). The best runs (individuals) are selected and mated for genes crossover. The next generation is then potentially better than the previous, and the procedure is repeated. The genes can be subjected to mutation as well. Finally, the offspring of a population does not significantly improve, and the search is terminated.

The obvious problem with evolutionary algorithms is that they require many function evaluations and, therefore, the computational power easily becomes a bottleneck. However, the cloud today provides affordable access to large computing power. An example of using GA for large-scale computing in the cloud is given in [20].

Our initial focus is on GAs, although most of the concept works with any algorithm that treats the objective function calculation as a black box. In the building domain optimization, which shares many similarities with the ship energy system optimization, GAs are by far the most popular approach [14].

Our work is related to the research of Xu et al. [22], who investigate the opportunities of cloud computing for simulation-based optimization. While they look at the issue from a conceptual and algorithm level, our focus is more on the practical challenges of using cloud computing tools and platforms for this task.

4 FRAMEWORK IMPLEMENTATION

Figure 1 presents the distributed components of the optimization framework. As mentioned, we focus on the methods that are well suited for parallel computation. Accordingly, the framework is for running simulations in parallel; it could also be used for other purposes than optimization, e.g., sensitivity analysis or uncertainty quantification. For this discussion, we assume that we have an optimization algorithm that needs lots of simulations that compute the objective function; this is the main computational effort and everything else is negligible. The algorithm is such that the simulations can be executed in parallel batches of dozens at a time (the inputs to each simulation do not depend on the results of others in the same batch, only on previous batches). As part of our sub-hypothesis 3, we assume that the overhead of sending simulation inputs and results over the network is small compared to the execution time of the simulation itself.

To execute the simulations, a user of our framework needs to acquire a Kubernetes cluster. They are now available from major cloud providers. Some preparations are required for the cluster: notably Helm (a Kubernetes package manager) and an ingress server must be installed. There is a bit more to it than that but it needs to be done once per cluster, not for every model deployed.

The user then packages the simulation model as a Docker image. The image includes the server side software of our framework, typically as a base image, the simulator, any model data, and the implementation of a simple Python interface that the framework uses to access the model. The image is pushed to a container registry so that the cluster can later retrieve it from there. Then a simulation service cluster application is deployed with Helm. In simple cases all its components can use the same Docker image, passed as a parameter to the Helm chart. The number of workers is also specified as a parameter and should match the number of parallel simulations expected (it can also be adjusted later). Upon deployment a release name is assigned.

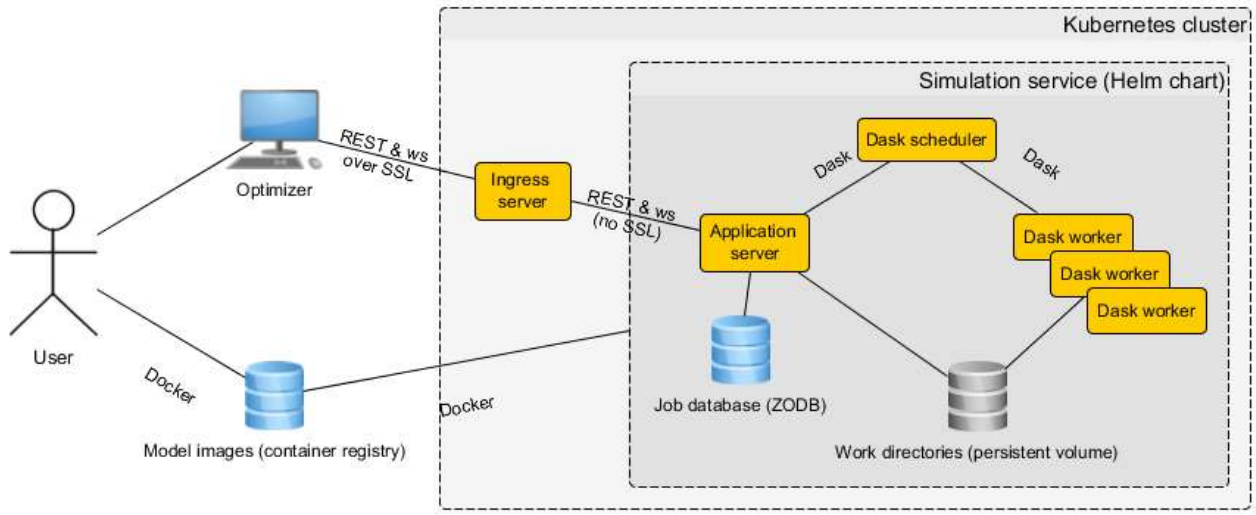

Figure 1. The components of the optimization framework.

The optimization software needs an evaluation plugin that can act as a client to the simulation service. Based on our work in an earlier project [23], we have implemented such a plugin for Opt4J[1], an optimization framework for Java. The evaluator is pointed to an URL derived from the simulation service release name. The services are model-specific and there can be several deployed on the same cluster. They are all behind the same ingress server, whose address is in the URL, and are distinguished by the beginning of the path. The client must also be provided with the authentication tokens that the simulation service was configured to require (we use basic HTTP authentication over SSL). The optimization software may run on the user's workstation or it can be deployed into the cloud; it only needs a HTTPS connection to the ingress server (preferably also WebSocket). Initially, we are working with the NSGA-II algorithm [21].

An optimization problem is then defined: the optimizer evaluates solution candidates by computing simulation input parameters from the decision variables, posting them to the simulation service, which creates a simulation job there, waiting for the job to complete, fetching its results and computing the optimization objectives from them. The ingress server proxies traffic to the appropriate application server, and also handles security. Most traffic is JSON data over HTTP but the client also connects to the server over Socket.IO. The server emits a Socket.IO event whenever the status of a computation task changes. Socket.IO is a higher level protocol similar to WebSocket and usually transmitted over it, which allows the client to wait without polling. However, if WebSocket is unavailable, Socket.IO automatically falls back to polling over HTTP.

The simulation service is written in Python; the web application is based on the Flask framework and distributes the execution of posted simulation jobs with the Dask library. Job inputs, state and results are stored in ZODB, a Python object database. Because simulators often produce logs or other files useful for debugging, we also have a persistent volume for them. It is shared by the application server and the workers: each job is assigned a directory where the worker may write whatever it sees fit, and the server makes those files available over the Internet. This is only for debugging; the application server does not attempt to make any sense of the data. Actual simulation results are returned via the model API, stored in the database and served to clients in JSON form.

Apart from the persistent volume, all communication inside the simulation service application is performed with the Dask library. The model interface consists of a Python function that receives the posted job inputs as parameters and returns a Dask task graph for computing the results. In simple cases the user would write a function that computes the results and decorate it with @dask.delayed. However, the interface also supports cases where each job parallelizes further. E.g., some vessel system is optimized to perform well under different sea and weather conditions, thus for each design candidate multiple scenarios are simulated with additional input data representing the different conditions. These simulations can be executed in parallel and the results are then combined by averaging or whatever is appropriate. The model function can return a more complex task graph that represents this, allowing Dask to distribute the simulations.

Once the optimization run has finished and the user decides that the data stored in the simulation service are no longer needed (because the interesting parts have been fetched), the service can be deleted with Helm. This stops all components and releases all storage.

5 CASE STUDY

Our first industrial case deals with a waste heat recovery concept, which design includes numerous different options in a modern cruising ship. The simulation software in this study is Apros (www.Apros.fi), which is a tool for full-scale modelling and dynamic simulation of

[1] http://opt4j.sourceforge.net/

industrial processes. Apros provides easy user interface for configuring and running the simulation models, efficient solution algorithms and model libraries for full-scale modelling and dynamic simulation of process, automation and electrical systems. See [8] for more information on the simulation environment and for an example application case.

Before this optimization case study, the simulation model was used in an engineering project within the shipyard company. Their modelling activities can be roughly divided into two classes with respect to the modelling principles: I) concept level models of the ship energy system, and II) rigorous thermal-hydraulic models of the energy systems. Typically, the simulation time step length in the former case is in minutes, while the latter case uses a time step around 0.2 seconds.

In a typical simulation, the model mimics the ship's operation over a certain period, e.g. a day, a month or a year, depending on the objectives and model's level of details (simulation speed). The simulation takes use of the ship's planned route, fulfilling and experiencing the requirements and conditions that prevail, depending on the time of the day and location within the route. For example, the location on the route may directly determine the vessel speed, while some energy consumption characteristics are derived from the time of the day and weather conditions, such as need for heating, cooling, lighting, etc. Important aspect during the simulation executions is to recognize and respect the prevailing limitations, caused by e.g. time constants in transients, realistic control modes and available system resources. Also, identifying the limiting components is valuable information for the design engineers. A simple example of a cost function is the cumulative fuel consumption.

Basically, both model classes mentioned are equally suitable for optimization with our framework. However, the related design (i.e. optimization) questions often differ considerably. It is typical in the concept level (class I) models that the alternatives are not just different model parameter values, but they address to various superstructures, i.e. alternative types and configuration of the process equipment. This potentially needs special arrangements within the model, or creating each superstructure ad-hoc before each simulation run. In fact, this is an observation against the original idea of taking out-of-a-box marine simulators and exploiting them. The models tend to need some extra attention before deployed to the optimization. For this reason, we have started the optimization experiments with the class II models.

6 LESSONS LEARNED

Table 1 summarizes how our experience supports or refutes the hypotheses we presented in Introduction.

In this project, we investigate the opportunities by the cloud computing, but it is worth mentioning, that the simulation service is capable to work in or outside the cloud. At simplest all its components can be executed locally in the same computer with the client; of course there is no distributed computation[2] then, so such a configuration is useful mainly for testing. For that it is useful though, as the working in the cloud adds several layers of complexity. Dask supports several scheduling back ends, e.g., SLURM, which is common in high performance computing. It also supports Kubernetes directly; one could run the application server on a local host and the workers in the cluster. However, Dask internal communication on the larger Internet would cause security concerns (e.g., it would be blocked by firewalls). HTTP and WebSocket should usually get through and have well established authentication mechanisms. Thus, it is most robust to deploy the application server into the cluster, keeping Dask communication in the cluster private network. That is actually why we implemented this as a web application; that and because we wanted to support optimization software written in different languages (so the optimizer cannot use Dask directly).

The cloud deployment was tested on Azure and initially also on Minikube (before security was implemented). We have tried to keep everything as independent of the cloud provider details as possible, just plain Kubernetes and Helm. Unfortunately, there are some provider specific details. In particular, the Azure Kubernetes Service must normally be created for a specified number of virtual machines (VM), and those VMs add to your bill per second whether you use them or not. You can explicitly scale the cluster but that is Azure-specific, bothersome and slow. There is a bridge called Virtual Kubelet, which should allow deploying with Kubernetes to Azure Container Instances, where containers are executed without dedicated (i.e., your) VMs underneath and you are only billed per computation time and storage that your containers use. Of course we wanted to use that. Unfortunately, we have not gotten it to work with the shared persistent volume. It is a documented limitation of ACI with Windows containers, but apparently a bug in Virtual Kubelet that it does not work with Linux either.

Transmitting numerical data over HTTP in JSON is hardly the most efficient protocol possible. However, it is usually not the bottleneck, if just the decision variables and objective values are transmitted. The inefficiency of sending numbers as text might become noticeable for a problem with a thousand decision variables. But with a GA problem of that size it is going to be the least of difficulties. Besides, an alternative data format could be easily added to the API. Also, the simulations often require a lot of static data that does not depend on the decision variables. Such data can either be packaged in the Docker image as files or be sent to the web application as default values, which are applied to every job.

Generally the Kubernetes and Docker scenes are somewhat Linux-centric. After all, Docker was first introduced for Linux. If Windows containers are needed for simulators that are not available for Linux, it requires more work to build them, because convenient base images are not that readily available.

[2] There is still parallelization for multiple cores

Table 1. Summary of the hypotheses and their support

Hypothesis	Experience	Future needs
1. Engineering problems that arise within the in-house simulation activities can commonly be converted into simulation-based optimization form, without reformulation of the simulation model (i.e. using them as black box simulators).	The interface between simulation and optimization needs effort. There is rather high effort to get the first model of a new type of simulator to run. The applicable range for size of problems not known yet. The superstructure optimization problems need special attention in the problem formulation.	Further validation needed
2. The black box simulators can be transferred and run in the cloud in the form of containers	Docker containers allow packaging simulations nicely. Additional effort on container management is needed because the container services in the cloud still require worrying about the VM layers	Container services need improvement to allow simply using containers the need to create VMs
3. Harnessing computing capacity in the cloud allows use of gradient-free optimization algorithms. Computation efficiency is achieved by parallel computing which the cloud provides on demand.	Practically infinite computing capacity is available, priced per capacity per time. Computation capacity can be fitted to the optimization task instead of fitting the task to available capacity. Capacity management is still somewhat cumbersome. Ideally it should be automated by the cloud provider and pricing should be per work, i.e., actually used rather than reserved capacity. Such services are emerging but not yet quite robust.	Better understanding of computational efforts for realistic energy optimization tasks. Methods to allocate optimal cloud resources for best price-performance ratio considering the different VM types available in the clouds.
4. Evolutionary optimization algorithms provide a means to find adequate solutions to the engineering problems.	Only tested with simple toy models so far.	Work with several industrial simulators required. Further validation needed.
5. Machine learning can help in directing the search of the optimization algorithm to improve its performance.	Not available yet	Part of our future work
6. The framework should easily allow changing the simulation model or the simulator, goals and constraints of the optimization, and the optimization algorithm. It allows the use of simulators built for different operating system (especially Windows and Linux).	Simple API defined between optimizer and simulator, but more versatile interface is needed. Although cloud providers somewhat support Windows containers, Linux containers are more popular and better supported. The current workflow is manual and clumsy. There seems to be considerable work for introducing a new simulator in the framework, for various reasons.	The interfaces need to be improved. The workflow needs consideration and probably tool support. More experimenting with different simulators needed.
7. The optimization framework promotes the optimization-as-a-service approach.	Not available yet	Technical, business model, and privacy issues need further work

7 CONCLUSIONS

In this paper we shared our qualitative experiences in developing a simulation-based optimization system using public cloud computing services. The aim is to use the system for solving optimization problems that arise with different maritime simulators, especially those involving dynamic behavior of a system. The present results are based on our early work and experiences with toy problems, while we are simultaneously preparing the industrial cases too. However, we feel that the results provide indications and insights about what cloud computing enables in this area and what kind of practical difficulties need to be dealt with.

As the summary in Table 1 indicates, there is plenty of future work to be done to make daily use of simulation-based optimization in the cloud easy. However, our early results indicate that this is a promising direction. There are problems, but they seem possible to solve with enough focused effort. The future will show how much we need to compromise our original objectives.

Our future work continues by evaluating the framework with industrial cases that deploy different simulation platforms, and by collecting experiences and numerical metrics on the performance. This involves reviewing our architectural choices and comparing with related, published approaches. We will investigate different optimization algorithm variants and their applicability to the cloud use. Flexibility to tackle different kinds of engineering problems is the key for success. Therefore, we will investigate, how agile our approach is and how it generalizes to different simulators. We want to find out the type and scope of optimization problems that best fit this approach, and means to cope with the others as well.

ACKNOWLEDGMENTS

This work belongs to the Task 2.3 in the INTENS research project, jointly funded by Business Finland's Arctic Seas program and the INTENS consortium (Wärtsilä Finland Oy, NAPA Oy, Meyer Turku Oy, Dinex Ecocat Oy, Deltamarin Oy, Vahterus Oy, Protacon Technologies Oy, Parker Hannifin Manufacturing Finland Oy, JTK Power Oy, 3D Studio Blomberg Ab, Jeppo Biogas Ab, Visorc Oy, Tallink Silja Oy, NLC Ferry Ab Oy, Aalto University, Lappeenranta University of Technology, University of Vaasa, Åbo Akademi University and VTT Technical Research Centre of Finland Ltd), which are gratefully acknowledged.

REFERENCES

[1] Boschert S, Rosen R. Digital twin—the simulation aspect. Mechatron. Futur., Springer; 2016, p. 59–74.

[2] Gosavi A. Simulation-Based Optimization. vol. 55. 2015. doi:10.1007/978-1-4899-7491-4.

[3] April J, Glover F, Kelly JP, Laguna M. Simulation-based optimization: practical introduction to simulation optimization. Proc. 35th Conf. Winter Simul. Driv. Innov., 2003, p. 71–8.

[4] Carson Y, Maria A. Simulation Optimization: Methods and Applications. Proc. 29th Conf. Winter Simul., 1997, p. 118–26. doi:10.1145/268437.268460.

[5] Andradóttir S. Simulation optimization. Handb Simul Princ Methodol Adv Appl Pract 1998:307–33.

[6] Fu MC, Glover FW, April J. Simulation optimization: a review, new developments, and applications. Proc. 37th Conf. Winter Simul., 2005, p. 83–95.

[7] Baldi F. Modelling, analysis and optimisation of ship energy systems. Chalmers University of Technology; 2016.

[8] Lepistö V, Lappalainen J, Sillanpää K, Ahtila P. Dynamic process simulation promotes energy efficient ship design. Ocean Eng 2016;111:43–55. doi:10.1016/j.oceaneng.2015.10.043.

[9] Zou G, Kinnunen A, Tervo K, Elg M, Tammi K, Kovanen P. Modeling ship energy flow with multi-domain simulation. 27th CIMAC World Congr., Shanghai: 2013.

[10] Lampe J, Rüde E, Papadopoulos Y, Kabir S. Model-based assessment of energy-efficiency, dependability, and cost-effectiveness of waste heat recovery systems onboard ship. Ocean Eng 2018;157:234–50. doi:10.1016/j.oceaneng.2018.03.062.

[11] Pedersen N, Madsen J, Vejlgaard-Laursen M. Co-simulation of distributed engine control system and network model using FMI & SCNSL. IFAC-PapersOnLine 2015;28:261–6. doi:10.1016/j.ifacol.2015.10.290.

[12] Yang MH, Yeh RH. Thermodynamic and economic performances optimization of an organic Rankine cycle system utilizing exhaust gas of a large marine diesel engine. Energy 2015;82:256–68. doi:10.1016/j.apenergy.2015.03.083.

[13] Ho Y-C, Zhao Q-C, Jia Q-S. Ordinal optimization: Soft optimization for hard problems. Springer Science & Business Media; 2008.

[14] Nguyen A-T, Reiter S, Rigo P. A review on simulation-based optimization methods applied to building performance analysis. Appl Energy 2014;113:1043–58. doi:10.1016/j.apenergy.2013.08.061.

[15] Trivyza NL, Rentizelas A, Theotokatos G. A novel multi-objective decision support method for ship energy systems synthesis to enhance sustainability. Energy Convers Manag 2018;168:128–49. doi:10.1016/j.enconman.2018.04.020.

[16] Sakalis GN, Frangopoulos CA. Intertemporal optimization of synthesis, design and operation of integrated energy systems of ships: General method and application on a system with Diesel main engines. Appl Energy 2018;226:991–1008. doi:10.1016/j.apenergy.2018.06.061.

[17] Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: a survey. J Marchine Learn Res 2018;18:1–43.

[18] Antoulas AC, Sorensen DC, Gugercin S. A survey of model reduction methods for large-scale systems. Contemp Math 2001;280:193–219.

[19] Cozad A, Sahinidis N V., Miller DC. Learning surrogate models for simulation-based optimization. AIChE J 2014;60:2211–27. doi:10.1002/aic.14418.

[20] Ding J, Yang S. Classification rules mining model with genetic algorithm in cloud computing. Int J Comput Appl 2012;48:24–32.

[21] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6:182–97.

[22] Xu J, Huang E, Chen C-H, Lee LH. 2nd Reading Simulation Optimization: A Review and Exploration in the New Era of Cloud Computing and Big Data 2nd Reading. Asia-Pacific J Oper Res 2015;32:1550019. doi:10.1142/S0217595915500190.

[23] Pardo N, Nadler F, Margueritte C, Kelly B. CITYOPT -- Holistic simulation and optimisation of energy in smart cities -- Vienna study case. J Environ Sci 2015;4.